REPORT DOCUMENTATION PAGE		Form Approved: OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	Final Report	Jan. 2003 – Dec. 2005

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Management of Test Complexity for Emerging Safety Critical Control Systems Program	F49620-03-C-0025
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Dale W. Boren, Embedded Software Engineer, Sr. Staff.	
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Lockheed Martin Aeronautics Company NUMBER Advanced Development Programs P.O. Box 748 Fort Worth, TX 76101	FZM-9204

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/NM 4015 Wilson Blvd. Room 713 Arlington VA 22203	Lt Col. Sharon Heise, USAF, PhD Acting Director, Mathematics & Space Air Force Office of Scientific Research 875 Randolph Street, Ste. 324, Rm 3112 Arlington, VA 22203	AFOSR ~~REPORT~~ AFRL-SR-AR-TR-06-0393

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release: distribution is unlimited.

13. SUPPLEMENTARY NOTES:

14. ABSTRACT

Future safety-critical flight control systems will contain sophisticated software algorithms with advanced functionality to enable autonomous operations. However, significant increases in complexity and volume of critical functions will excessively challenge current Verification and Validation (V&V) practices. Therefore, reducing and/or managing the complexity of systems has a major impact upon the cost and schedule of development, verification and maintenance of current and future safety-critical flight control systems.

The current research first seeks to establish and identify sources of system complexity related to the logical make up of the system. A continuous logic (i.e., Logical Matrix Algebra, LMA) was applied first to the entire system as a Safety Assurance Monitor (SAM), and then at a functional level resulting in a Correct-by-Construction Artificial Neural Network, (CCANN). The feasibility of applying these new tools was considered and conclusions drawn from an overall process view to suggest where such tools, may be applied.

15. SUBJECT TERMS
Complexity Management, Safety Critical, Correct-by-Construction Artificial Neural Network,

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dale W. Boren
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified	UL	60	19b. TELEPHONE NUMBER (include area code) 817-935-5843

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

FINAL REPORT

Management of Test Complexity for Emerging Safety Critical Control Systems Program

Contract No.: F49620-03-C-0025

SOW Para.: 3.4.1a DID: DI-MGMT-80711A

Document No. : FZM-9204

Issue Date: May 26, 2006

Prepared and Approved by	Date
/s/ Dale W. Boren	May 26, 2006
Dale W. Boren	
Title: Embedded Software Engineer Senior Staff	

Prepared for

Air Force Office of Scientific Research - AFOSR

Prepared by

LOCKHEED MARTIN CORPORATION
Lockheed Martin Aeronautics Company
Lockheed Blvd. Fort Worth, Texas 76010

REPORT DOCUMENTATION PAGE		Form Approved: OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	Final Report	Jan. 2003 – Dec. 2005

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Management of Test Complexity for Emerging Safety Critical Control Systems Program	F49620-03-C-0025
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
Dale W. Boren, Embedded Software Engineer, Sr. Staff.	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Lockheed Martin Aeronautics Company NUMBER Advanced Development Programs P.O. Box 748 Fort Worth, TX 76101	FZM-9204

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/NM 4015 Wilson Blvd. Room 713 Arlington VA 22203	Lt Col. Sharon Heise, USAF, PhD Acting Director, Mathematics & Space Air Force Office of Scientific Research 875 Randolph Street, Ste. 324, Rm 3112 Arlington, VA 22203	AFOSR
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release: distribution is unlimited.

13. SUPPLEMENTARY NOTES: Copyright 2006 by Lockheed Corp.

14. ABSTRACT

Future safety-critical flight control systems will contain sophisticated software algorithms with advanced functionality to enable autonomous operations. However, significant increases in complexity and volume of critical functions will excessively challenge current Verification and Validation (V&V) practices. Therefore, reducing and/or managing the complexity of systems has a major impact upon the cost and schedule of development, verification and maintenance of current and future safety-critical flight control systems.

The current research first seeks to establish and identify sources of system complexity related to the logical make up of the system. A continuous logic (i.e., Logical Matrix Algebra, LMA) was applied first to the entire system as a Safety Assurance Monitor (SAM), and then at a functional level resulting in a Correct-by-Construction Artificial Neural Network, (CCANN). The feasibility of applying these new tools was considered and conclusions drawn from an overall process view to suggest where such tools, may be applied.

15. SUBJECT TERMS
Complexity Management, Safety Critical, Correct-by-Construction Artificial Neural Network,

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dale W. Boren
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified	UL	60	19b. TELEPHONE NUMBER (include area code) 817-935-5843

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

REVISION STATUS

REVISION	DATE	COMMENTS
0	May 18, 2006	Initial LMAero Format
1	May 29, 2006	AFOSR Format

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Page

1. ABSTRACT

Future safety-critical flight control systems will contain sophisticated software algorithms with advanced functionality to enable autonomous operations. Transition and implementation of these emerging algorithms will ultimately depend on affordable verification and validation (V&V) of prescribed safety and reliability for fight certification. However, significant increases in complexity and volume of critical functions will excessively challenge current V&V practices. V&V of safety-critical flight control systems is already time consuming and costly, generally representing as much as one third the overall cost of the system. System complexity makes V&V hard. Therefore, reducing and/or managing the complexity of systems has a major impact upon the cost and schedule of development, verification and maintenance of current and future safety-critical flight control systems.

This research is directed at analysing the development of safety-critical flight control systems and determining what, if any fundamental methods, processes and practices may be implemented to reduce and manage complexity in current and future safety-critical flight control systems. The primary focus is mathematical, starting with a formal representation of the customer's requirements, fleshing out those requirements to make them complete, consistent and verifiable, to modelling the system that those specifications define and implementing this design in a verifiably safety-critical way. Many issues enter into this overall process, not all are technical, but all have significant effect on the growth of system complexity throughout its development process. At each stage formal logic is applied, the requirements may be written as truth tables, the system modelled by an automated tool based upon a formal language, software coded in a computer language on an operating system loaded on hardware that implements the

code in registers made of logic gates. Each stage of system development adds layer upon layer of complexity to the end result. Opportunities to prevent complexity growth (i.e., simplify) must be applied at each phase and at the interfaces between the phases of development.

Nature's design is elegant. A common fly has impressive flight performance capabilities, but requires only six layers of networked neurons from sensor to actuator. Nature, therefore suggests that there is room for reducing complexity of safety-critical flight control systems. It also suggests that our current approach to system development has much room for improvement. Our research builds upon previous research in developing an extended logic over the real numbers. This logical framework is called Logical Matrix Algebra (LMA) [1] and is closely related to neural networks. The present research seeks ways to apply this algebra as a fundamental dynamic system representation to simplify the design of safety-critical systems.

The current research first seeks to establish and identify sources of system complexity related to the logical make up of the system. Redundancies inherent in the design of such systems, was demonstrated. Using a continuous logic (LMA), over lapping system stability and coverage parameters were analysed for possible relationship to reduce system complexity. System architecture was considered and practical implementations of continuous logic were applied. This logic was first applied to the entire system (i.e., Safety Assurance Monitor (SAM)), and then at a functional level (i.e., Correct-by-Construction Artificial Neural Networks (CCANNs)). An introduction to this algebra is given in the appendix. The feasibility of applying these new tools as well as existing tools, techniques and strategies was considered and conclusions drawn from an overall process view to suggest where such tools, techniques and strategies may be applied.

2. OBJECTIVE/GOALS

2.1 Management of Test Complexity For Emerging Safety-Critical Control Systems

Future safety-critical flight control systems will contain sophisticated software algorithms with advanced functionality to enable autonomous operations. Transition and implementation of these emerging algorithms will ultimately depend on affordable verification and validation (V&V) of prescribed safety and reliability for fight certification. However, significant increases in complexity and volume of critical functions will excessively challenge current V&V practices (Figure 1). V&V of safety-critical flight control systems is already time consuming and costly, generally representing as much as one third the overall cost of the system (Figure 2). System complexity makes V&V hard. Therefore, reducing and/or managing the complexity of systems has a major impact upon the cost and schedule of development, verification and maintenance of current and future safety-critical flight control systems.

Next-generation unmanned air vehicles (UAVs) and unmanned space vehicles will require advanced safety-critical system attributes to enable safe autonomous operations. Adaptation, learning, optimization, and prediction will provide necessary intelligence for on-line decision-making, reasoning, and cooperation. Optimal architectures may include integration of functions with various levels of criticality amongst physically distributed processors. These attributes will increase the size and complexity of control systems beyond the capability of current V&V practices (Figure 1).

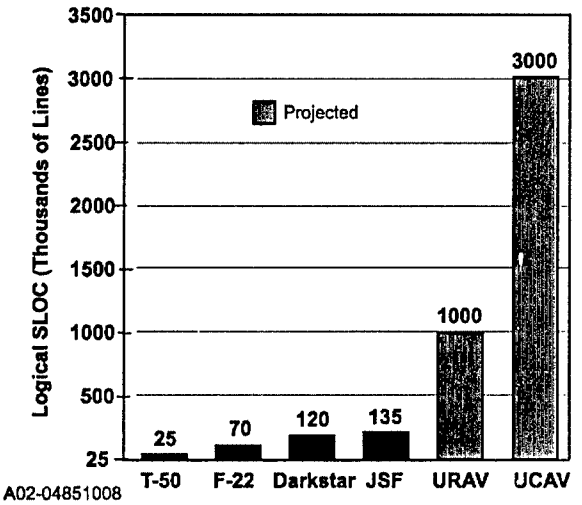


Figure 1. Exponential Growth of Flight-Safety-Critical System Complexity

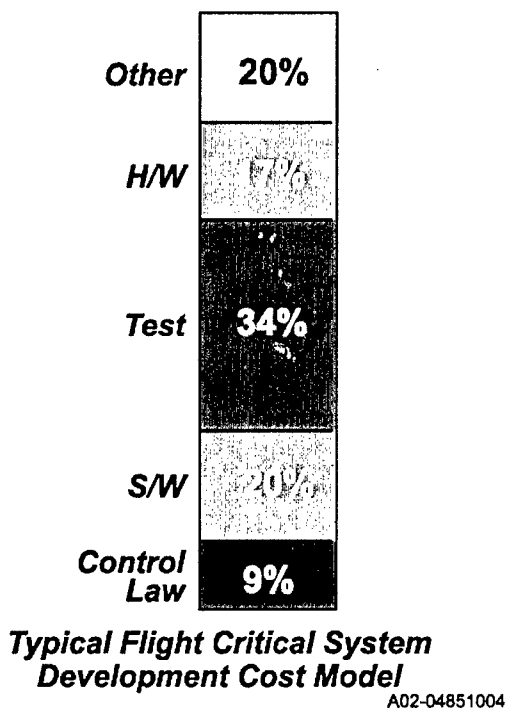


Figure 2. Costs of Design and Testing Dominate Current Flight-Safety-Critical Systems

2.2 Purpose

Our primary program objective is to conduct fundamental dynamical system and control theoretical research and development in the area of managing the complexity of highly sophisticated and emerging safety-critical control system. "Management" must include prevention particularly in managing complexity growth throughout development and the life cycle of safety-critical air vehicle system. Complexity is fundamentally a sizing problem. Preventing system growth in size (i.e., complexity) will assist existing and future management methodologies for such systems. Complexity management also requires a balance between component level simplification and system level optimization. Complexity growth prevention and balancing system architecture are addressed through fundamental mathematical concepts, revising when and how vital calculations are performed. Just as a specific system's performance may be improved by simply changing the way it is used, so altering when and how specific development tasks, such as design calculations etc., are performed will enhance the development process itself.

2.3 Scope

To keep the scope of the research within funding and other resource limits our focus is upon complexity sources that increase the numerical growth of test cases as an indication of the underlying mathematical (i.e., combinatorial) complexity of the system. This approach emphasizes individual engineering skills development in the spirit of "Best Practices" which can be implemented throughout the system development cycle. Further, this approach allows immediate implementation of separate concepts and techniques as appropriate to existing programs.

2.4 Report Development

The first step in mitigating the problem of system complexity is to identify the primary sources or root causes of system complexity. After identification of sources and/or causes a study within the limits of the resources available for the research is required. Much has been done both in industry and academia to identify and address sources of system complexity with recommendations for better business practices, policies, procedures etc. However, there is no single point source or root-cause of complexity, hence no silver bullet can be expected to resolve the overall problem of system complexity. Many efforts in managing system complexity have resulted in methods and automated tools based upon formal methods. These methods and tools have had varying degrees of success depending on when, where and how they are implemented. As a general rule, the earlier the implementation and the better their integration throughout the entire development process, the greater the benefits resulting from these methods and/or tools.

One strategy to managing system complexity is to mitigate its growth throughout the development process. Prevention, many times, is as or more powerful than a cure. A complexity growth profile representing system complexity growth at each stage of the development process is helpful in identifying development phases that contribute most to overall system complexity.

Figure 3 is based upon the number of design (i.e., modelling) blocks and source lines of code (SLOC) implementing the system design. Clearly, at each stage of development the system's complexity increases showing the adage "the devil is in the details" to be true. Complexity increases dramatically as design details flesh out requirements and then again as designs are implemented in software and/or hardware.

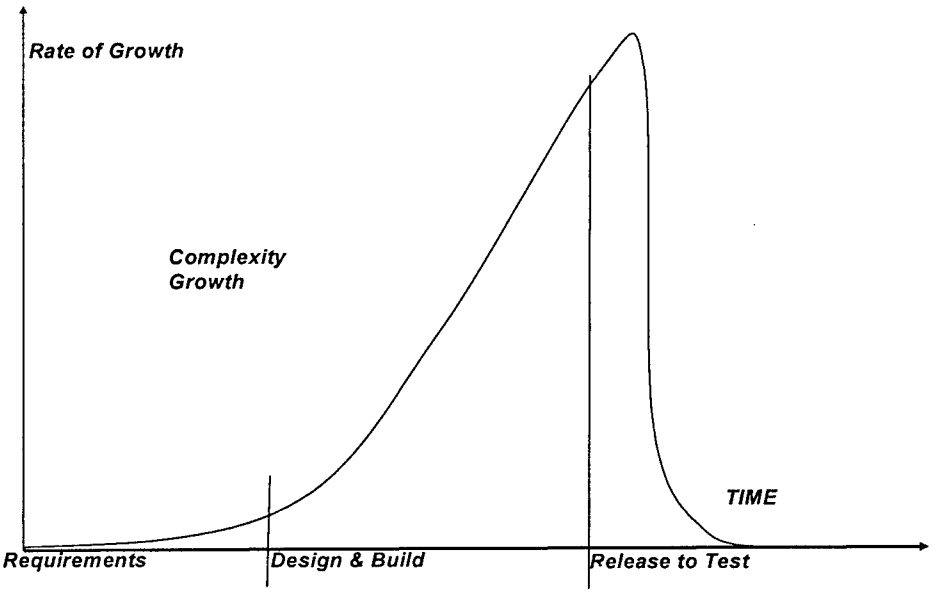


Figure 3. Growth of Complexity throughout the development process

If requirements can be directly implemented and/or the designs derived from the initial requirements can be simplified significantly, then the growth of system complexity is greatly reduced. Much has been done in this direction. Requirements are often written in a formal language of some type (i.e., UML, Rational, etc.), design is often carried out with a formal language based tool or tools that approach this kind of formalism (i.e., SystemBuild, Simulink etc.) and final software implementation has become the production of automated code generation. Formal language tools allow verification of the logic for completeness and/or complexity at the earliest stages of development. However, completeness and consistency is not logical reduction. Requirements, design function block models and source code implementation are not currently subject to systematic (i.e., formal) logical reduction other than by the skill of the engineer performing the task. The point is, if requirements can be automatically checked for completeness and correctness, then the system logic should also be systematically reducible to

lower, if not minimum terms. Design engineers are left the discretion to apply such reductions or not, perhaps electing for more efficient system integration. Recall that "optimal" may not be minimal or least costly. As the system proceeds through development and becomes more complex, the ability to reduce the logic of the overall system becomes more and more difficult. However, initial efforts early on yield benefits through prevention of complexity growth. The savings is admittedly difficult to quantity or measure.

Task_1 of our approach seeks to measure or at least to demonstrate that system complexity can be reduced at the modeling level, if not also at the requirements level. Task_1 is to develop a representative baseline control model and analysis this model for opportunities to reduce the logic of the system. Observations about current safety-critical flight control system development leads to the assumption that hidden redundancies may exist yielding opportunities for system simplification. (1) System Development Engineering Structure where separate teams of engineers develop individual components of the system for later integration. Systems engineering seeks an architecture of the overall system to reduce its cost and complexity, nevertheless this necessary system development engineering structure leads to logical inefficiencies, (2) Flight control systems are built around design points throughout the flight envelope. Very similar controllers are built to perform optimally about specific design points and are merged to give the air vehicle control between those design points. (3) Working back from the implementation of the software design it is seen that code can be viewed as interleaved guard and functional statements wrapped in flow control statements. Whether code is implemented using a block tree structure (i.e., a structure of subsystems) or inline makes a difference as to the number of test cases required to verify it. This is similar to summing 7, 4 times which may be

implemented as 7 + 7 + 7 + 7 or 4*7. The result is the same, but the implementation and verification is different (i.e., three sums vs. one product).

The above example illustrates that complexity has a subjective component. Some would view one implication less complex than the other. However, form a computational point of view, the number of operations or the speed of those operations becomes an objective measure.

Task_2: Stability and Coverage Relationships Development is based upon the observation that both guard statements and the functions they enable use many of the same variables. This implies that if a relationship can be found linking the two (i.e., logical, functional) operations a more general form of mathematics would allow the combining of nearly similar controllers from different design points. Evidence suggests that such a generalization would be related to neural network like structures. A recently developed mathematical representation of a continuous logic called Logical Matrix Algebra was used in an attempt to combine guard logic and functional operations in a signal network notation.

Task_3: Testable Control System Architecture Development and Analysis seeks to integrate lessons learned from the previous two tasks at the systems architectural level. Opportunities for architectural optimization were considered. As with task 2, implementation of this analysis as applied to the architecture of safety-critical systems will be address later in this document.

Finally, Task_4: Test Case Reduction Feasibility Assessment reviews what progress can be made in reducing the overall complexity of safety-critical flight control systems in light of the above strategies and/or methodologies.

3. APPROACH

3.1 Methods

Many methods including Formal Methods, Modeling, and Review of legacy programs for identifying sources of system complexity have been used to compile this report. Those methods directly applicable to the scope of this research are given below.

3.1.1 Formal Methods

Formal Methods are widely known and used in many industries and are the basis for many applications, tools and processes applied in system development. Our primary interest in Formal Methods is their use as the basis for modeling systems. Formal methods based upon formal languages are particularly important in modeling safety-critical systems. They allow for rigorous analysis and control of model construction. The primary formal method/language applied in this research is known as Logical Matrix Algebra (LMA). This is an extension of binary logic to the real numbers. It only requires the added definition of a saturation function yielding a saturated sums and products representation of 'OR' and 'AND,' the definition of the 'NOT' operator is naturally defined over the real numbers as used in binary logic (i.e., $x^C = 1 - x$). The definitions and development of Logical Matrix Algebra is left to the appendix of this report to avoid being side tracked from the primary interest of the report which is safety-critical system complexity management. The reader is referred to the appendix for an introduction.

3.1.2 Modeling

Modelling is the primary tool for managing system complexity. Models are representations of systems that carry along characteristics necessary to control and analyse the system while minimizing or hiding other characteristics that would otherwise obscure

understanding. As stated above we will use automated tools and formal modelling languages to build representative models of current safety-critical flight control systems. These tools and language will allow us to rigorously analyse these models for opportunities to avoid and manage the root causes of complexity within these safety-critical systems.

3.1.2.1 Models as Representations of Systems

Models are representations of systems and not the systems themselves. No single model may be said to completely represent a physical system, rather it represents those features (i.e., parameters) of the system that the engineer desired to manipulate to achieve some specific goal. Different types of models are used as appropriate. Transformations between system representations such as differential to Laplace (i.e., time domain to frequency domain) simplifies system analysis by bring forward to features of the system most relevant to the current analysis. In the time to frequency transformation instance, table look-up replaces much of the time calculation in the analysis of the system.

There are various types of modelling techniques, such as, diagrams (graphs), tables, decision trees, function blocks, network including fuzzy systems etc. Each model type has its strengths and weaknesses. For example, graphs are useful, but limited to small systems. It is often not practical to change or redraw a graph even with automated assistance. Viewing a graph of the whole system is rarely feasible. Tables allow for changes to be made more easily than graphs. Tables are very orderly; however, they are more abstract and suffer from human inability to view the system as a whole. The size of the system in every case is a challenge to any modelling strategy. The machine representation of many of these modelling strategies reduces to matrices. Our research focuses upon using the properties of matrices in a formalized language to support safety-critical features of the modelled system.

3.1.2.2 Models verses Requirement Specifications

Strictly speaking, models are not requirement specifications as some have suggested. Models represent a specific instance of a requirement's specification, they are a design tool answering 'HOW' rather than 'WHAT' the system is to do. Graphical requirements tools usually based upon UML which in some instances may become executable and graphical modelling tools are blurring the line between requirements and design. This blurring of requirements and design is primarily positive. However, it also yields some negative effects from a systems verification point of view. Positive effects include the possibility of early model checking for consistency, completeness and simulation. Negative effects include yielding to the temptation of avoiding laying a solid foundation of rigorous requirements and throwing function blocks together. Patching functional blocks together and then run a quick simulation with standard inputs to show an expected result is not adequate for safety-critical system development. A through analysis of the model is required. The tools for such an analysis currently exist, but lacking a foundation of rigorous written requirements these tools are often not used. The reasons for this include customer expectations reflected in contractual agreements. More on this issue later. For now it is only necessary to point out that sources of complexity within the system may stem from how the system is represented, when tools for analysing the system are applied or not applied and how the effects of these decisions are handled throughout the rest of the development process. In general, complete requirements analysis requires knowledge of all input signal range constraints, unlike simulations which simply apply specific inputs usually representing expected system inputs with perhaps some known error conditions or scenarios. Range constraints are pre conditions as expected outputs are post conditions both of which are necessary for verification of a systems performance. Once this pre and post conditional

information is available, automated test vector generation, model checking and some theorem proving may occur. Importantly, this information is available, if only in the engineer's head, at the earliest stages of system development. Note: absolute numbers are not required at first, initial guesses are allowed and will become more refined as the iterative development process proceeds. Though Model based development and its related automated tools are valuables engineering tools it appears that they do not require design engineers to think in terms of the testability of their design. Our research into managing the complexity of safety-critical system development incorporates the valuable lesson of "Beginning with the end in mind." In this case, the 'end' is verification and validation of the safety-critical system. Prevention of ambiguous information and remaining within the scope of the original requirements is significant in preventing the growth of system complexity throughout the development process.

3.2 Assumptions

It is assumed that the reader is familiar with such tools as MathWork's Simulink or other similar 4^{th} generation modeling tools. Though not based upon an underlying formal language Simulink was a readily available resource and thus used as our modeling tool. Its primary limitation (i.e., lacking the ability to dynamically resize matrices) is shared with all such modeling tools currently available.

It is further assumed that the reader has some experience in the development and verification of safety-critical systems. Specifically, any experience with system verification using automated tools would be helpful, but not critical. The author has tried present result summaries in non-technical terms. It would be beneficial to have experience with flight controls systems. Else, it is expected that the reader can follow standard algebraic, matrix and vector mathematical notation. Logical Matrix Algebra (LMA) is developed in the appendix.

3.2.1 Sources of System Complexity

No attempt here is made to list all elements that contribute to system complexity. The following lists some of the significant contributors, particularly those our research hopes to impact.

3.2.1.1 Hardware Contributions to System Complexity

Hardware is a significant component of the system with several sources of system complexity. We will only reference a small number of them. In particular we have considered multiple redundant sensor information. Our research assumes that this information has been decoded, voted and otherwise manipulated such that the input signals to our representative system models are known verified signals values. We have also considered the system complexity do to actuator signal limiting devices and/or software. These however, will not be added to our representative system models as part of our goal is to eliminate such complexities by more adequately representing pre and post conditions of the functions leading to these actuator signals. One does not need to limit a signal whose constraints are well known, accounted for and controlled.

During processing computational complexity arises from more than just large numbers of system components it also has an element of uncertainty arising from how system components are combined. This can be illustrated by comparing functionally equivalent designs of a simple hardware circuit. For example, let a 7-segment display circuit have four binary inputs and seven binary outputs in a two layered AND/OR circuit. Using the least complex individual components (i.e., 2-input 'AND,' and 2-input 'OR' gates) the circuit requires 123 gates, 17 gates per output plus and additional four 'NOT' gates. We may reduce the number of gates required by using logic reduction. This reduces the total number of gates to only 46 (35: 'AND,' 7: 'OR' and 4:

'NOT'). The output remains the same, but our ability to evaluate the logic is slightly diminished as each term groups more inputs together. Allowing like components to drive multiple outputs reduces the number of required gates another 30%. Finally, additional savings in the number of components can be had if we allow for multiple layers of logic gates. This, however, causes each output's signal to have a different time delay, adding addition verification issues.

From a path coverage verification point of view, each reduction in the number of gates usually reduces the number of paths needing to verify the circuit. However, when gates are used to drive more than one output the number of paths may increase. Simply reducing the number of components in a system does not necessarily reduce the complexity of the system from a verification point of view.

With all of these sources that significantly impact the complexity of a system it is acknowledges that no one solution will fit all conditions, rather strategies implemented by combining different existing methods, products and/or newly developed ones will be key to managing over all system complexity.

3.2.1.2 Software Contributions to System Complexity

It is acknowledged that the sources of software complexity are numerable. For instance, software can be configured to run multiple ways on the same hardware taking advantage of the hardware's options for packaging and transmitting data. Additionally, a great deal of software complexity results from the incompleteness and/or inconsistency of the requirements, design model, code generator etc. that the software was derived from. Verification of as little as three command statements illustrates some of the issues, (the following is taken from the Professional Software Engineering Licensing exam). The exam asks, what is the fewest number of test cases required to achieve Multiple Condition / Decision Coverage (MC/DC) for the following,

A = 100;

If ((B > 50) AND (C < 300)), Then A = 1000;

If ((B > 10) AND (C > 100)), Then A = 10;

For this example, a minimum of eight test cases are required. The solution vectors are not unique, if we use boundary theory, one solution would be (B, C) = [(0, 0), (10^-, 300^+), (10^+, 100^+), (50^+, 300^+), (10^+, 0), (50^-, 100^-), (50^+, 0), (50^+, 100^-)], where N^- means: just less than N and N^+ means: just greater than the boundary N. It should be noted that there are at least three interpretations of how to implement the above relationships. Each interpretation differs in their assumed scoping (i.e., nesting) of the if/then statements. Further, there is an over lapping of the range space of the two if/then statements requiring consideration of the intended sequence of execution.

This example shows that even the simplest systems can add complexity by not stating enough specific information. The code is simple, but the context is ambiguous.

3.2.1.2.1 Key Observations

Our test vector generation tools rely on the fact that systems/software functional relationships are enabled by logical decisions [2]. That is, functions are interleaved within a logical decision structure. Requirements state 'What' and 'When' the system/software should do something. Design/Models and Implementation/Build give the details of 'How.' As noted above, these details represent the primary addition of complexity with in the development process. Though the functional components contribute to the decision structure via constraint conditions, the largest contributor to the complexity of the system software is usually the logical decision component. The MCDC testing criteria applies to this component. Looking more closely at the makeup of this decision component, we observe three related elements contribute to the systems

complexity space (1) the decision structure, (2) the decisions themselves, and (3) the conditions (i.e., inputs variables) and the constraints on the conditions. Figure 4 illustrates the complexity space and its relationship to the decision structure of a system.

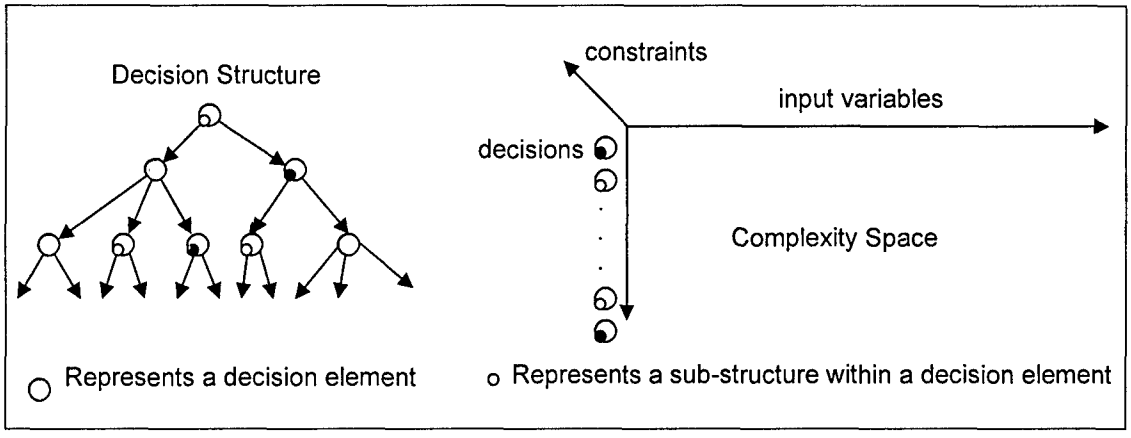


Figure 4. System/Software Decision Structure and related Complexity Space

To reduce the complexity space we may reduce any or all three elements. Fewer variables may imply less complicated decisions. Identifying redundant or equivalent variables would reduce the number of conditions needed to meet MCDC. Some variables may be reduced by stacking them. This, however, only creates another variable with increased constraints. Clearly, reducing the number of decisions simplifies the Decision Structure by cutting out nodes. Reducing the Complexity space by reducing decisions is analogous to finding the minimum number of logic gates needed to implement a seven-segment display. Decision constraints may be inherited from upper level decisions or imposed by the limits of the functions they enable. Therefore, it may be possible to derived constraints from the top down and/or the bottom up. Later analysis showed this approach to result in greater difficulty than anticipated.

3.2.1.3 Sources of Complexity during Software System Development

During the development process itself significant complexity can be added to the system. Some of these sources are outside the scope of this research, but should be mentioned as they have a significant impact upon the level of complexity of the system. These sources demonstrate that management of system complexity is not only a technical matter, but a development, engineering mind set, corporate cultural and contracting matter as well.

3.2.1.3.1 System Development

If we are to ever manage the growth of complexity throughout the development process, we must begin at the beginning with the top level requirements obtained directly from the customer. These requirements often change throughout development as the customer's priorities change. Nevertheless, obtaining a clear, complete and unambiguous picture of the desired result is the single most important step in successfully meeting the customer's expectations. "The Clearer the Picture, the Surer the Success." Most often these initial requirements are not analysed for testability, completeness or correctness which can lead to the most serious and costly development costs.

3.2.1.3.2 Engineering Mind Set

The engineer's mind-set while designing a system is primarily functional, that is, designers usually seek to build systems as a combination of functional blocks. Design tools support such a mind-set. However, verification of a system requires a different mind-set. Verification is based upon analysing pre and post conditions. These two mind-sets can be likened to writing a composition. When composing a document the author is thinking about what to say, not necessarily grammar and spelling. The process of composing requires toggling back and

forth between thinking about what to write and analysing the writing for correctness. The saying, "begin with the end in mind," applies here. The system designer would do well to place himself in the position of system verification engineer. One specific thing that a design engineer can do to quickly to enable the early analysis of his design, is to deliberately and clearly document the expected system input and output range values including units for each system functional blocks. These values are required by all the automated test case generations tools and are used during the critical and expensive verification phase of system development. It is noted that this activity would not take much time as the designer has all the information in his head. Like composition, system development is an iterative process. It is not necessary that the final input/output ranges be precise at first. An order of magnitude estimate is sufficient to allow these automated tools to check for consistency, correctness and run simulations. Automation makes this iterative process relatively painless. The design engineer has made these assumptions as a natural part of the design process, all that is needed is for him/her to document these assumptions so they may be inputted into the automated tools and passed down the development process.

Insuring that requirements are testable, complete and consistent requires no new technology (i.e., MIT, Systems and Software Consortium Inc. etc.) and is the single most valuable step in managing the growth of complexity in the developing system. It does this simply by preventing the addition of unnecessary complexity.

More than just checking consistency and completeness of requirement logic, these tools may perform boundary analysis, if given input domain, and output range values. It is currently feasible to run testability, completeness and consistency checks at every development phase using automated tools. It is now feasible to develop systems through a "correct-by-construction" system development process.

3.2.1.3.3 Corporate Culture

Corporate Culture includes both costumer's expectations as well as the supplier's commercial incentives. Some mention of the appropriate application of technology must be made. Throwing money at a problem does not guarantee that the problem will be solved or that the solution will be cost effective. As an example, one area in which the several branches of the US military are spending a lot of money with a degree of success, but at higher costs than is necessary is in the area of small and miniature UAV deployment. The connection with complexity management lays in the fact that miniaturization requires cramming all the intelligence, sensors and power supplies onto small platforms. This results in costly additional requirements that are difficult to meet. The adage, "Keep it Simple," prevents systems from becoming overly complicated to begin with. It has become clear that using a multitude of different types of UAVs adds unforseen complexity [4] not only to the systems themselves but to the operations of other systems they impact. Radio bandwidth is limited and being consumed rapidly. Little interoperability between systems makes their effectiveness less than optimal. One simple idea to use current technology as appropriate rather than expensive cutting edge technology would resolve a host of problems while accomplishing specific missions at low cost. However, it lacks the 'gee-whiz' factor, an artificial costumer expectation. A supposed limitation condemns it before any evaluation of merits is considered, a supplies bias towards cutting edge technologies. What is this simple idea? Apply an optical tether carrying both power as well as data to a semi-autonomous small/miniature air vehicle. Such a platform (i.e., a hovering uav) could be used to investigate and disarm improvised explosive devices (IED) at a safe distance, provide situational awareness for solders in congested urban environments peeking around corners, down hall ways and into high story windows, exploring and mapping caves and bunkers

where gps and control signals are not readily available etc. The on station time is unlimited because power is being pumped to the platform instead of being carried on board. Radio bandwidth is not consumed as a direct and secure channel is provided which makes the additional complexity of encrypting signals unnecessary. Significantly, costs as well as complexity are reduced because fewer elements need miniaturization and all other system elements are commercial off the shelf (COTS). The unwarranted concern of tether entanglement is countered by the appropriate use of such a device and the long history of tether use in cave diving etc. Thus, applying appropriate technologies go for the given task will go far to managing system complexity by preventing unnecessary requirements and constraints etc. Keep it simple.

3.2.1.3.4 Contracting Issues

Contracts are written as they have always been, with little consideration of how new technology can most effectively be used and measurably evaluated by contract milestones. Contracts reward companies for producing products not writing specifications and running preventative analysis. Engineers are constantly pushing to meet milestones that reflect measurable results, but do not prevent problems down stream. They are rewarded for immediate demonstration with little, usually no, incentive for prevention. It is observed that program contractual requirements do not reflect the importance of the essential steps of preventing system complexity. Thus, contractual limitations are a major factor in program cost overruns. No one is suggesting relaxing contractual requirements to demonstrate performance. Rather, it is suggested that contracts also include the long view of the development process. Again, "beginning with the end in mind," will aid in providing incentives to lay a solid foundation for product development. Contracts should reflect a balance between long term and short term quality and performance objectives.

4. PROGRESS

4.1 Baseline Control Model Development and Analysis – Task 1

A baseline or truth, representative control model was developed including gain switching and with deliberated redundant components. Legacy air vehicle control systems are designed about specific control points in their desired flight envelope. As the air vehicle transitions between these design points the control algorithm adjusts its parameters (i.e., gain, poles and/or zeros) accordingly. One question in redundancy reduction was whether nearly redundant feedback loops derived from adjacent design points can be safely reduced by increasing the distance between the design points. The idea was that automated test case generation might reveal domain to range mapping information at would lead to more efficient coverage using fewer design points. Our Goal was to guide the design and implementation of system/software by using the functional constraints (i.e., stability constraints). It was hoped that this would limit the number of coverage constraints on the system. Reducing the number of coverage constraints directly reduces the number of test cases needed to comply with the MCDC testing criteria.

A number of equivalent models representing the same functionality were generated. Hidden redundancies within a base model were manually removed and a generalized Logical Matrix Algebra (LMA) equivalent model created. Elements contributing to the generation of test cases were evaluated. Relationships between coverage constraints and dynamic stability constraints were determined as far as possible to identify critical test cases.

A review of LMAreo's System/Software development process was done with the view of determining where in the development process the majority of the complexity growth occurred. As shown in figure 3, the design and build stage of the development process is where the bulk of the systems complexity is added.

Current and future air and space systems are so large that no one engineer or group of engineers can perform all development tasks. This means that several groups of engineers must cooperate together to build these advanced systems. Communication and cooperation between these different groups has always been a significant part of the overall engineering process. The dispersion of tasks among groups of engineers and the overlapping nature of many of those tasks leads to inefficiencies and/or hidden redundancies. It was assumed that these inefficiencies and/or hidden redundancies lead directly to additional test cases. If true, it was assumed that the system/software could be altered to remove these additional test cases without diminishing its safety and/or performance.

We first created a small model with built in redundancies to demonstrate that hidden redundancies could lead to excess test cases. Using state of the art automatic test vector generation tools [2], we obtained the minimum number of test vectors required to completely evaluate the model. Here 'completely' means a sufficient number of test cases were generated to satisfy the Modified Decision Condition Coverage (MCDC) testing criteria. The model was then reduced by in-lining the subsystems. In-lining is a process that flattens the hierarchy of a model. All subsystems are brought up to a single level. From a coding perspective, in-lining inserts the actual code for a macro at every place the macro is called. It is not directly apparent, that in-lining should reduce the number of test cases needed to evaluate the system. We therefore, constructed a small demonstration model to evaluate this potential. This model showed that in-lining could reduce the minimum number of test cases necessary to meet the MDCD requirement. This verified our assumption that hidden redundancies may lead to additional test cases.

Next a reduced model was generated by hand. All models were built using MathWork's Simulink modelling tool. This reduced model performed the same functionality as the original model, but had a less complicated architecture. Using our tools, test cases were generated for the reduced model. Test vectors generated for the in-lined and the manually reduced model were the same. Finally, an equivalent model was created using Logical Matrix Algebra (LMA) [1] concepts and the Simulink tool. LMA enables the construction of a fixed neural network that directly relates the input/output data to the structure of the network. That is, we built the LMA network from the test vectors of the reduce model and this LMA network (LMAN) successfully approximated the reduced model. All test cases were verified on this network. We then altered the parameters of the LMA network and evaluated how these alterations would affect the network's ability to approximate the model.

In-lining, as well as, the reduced and LMA equivalent models reflect the variety of architectures that can produce the same result. This verified our second assumption that the system/software can be altered to remove these additional test cases without diminishing its safety and/or performance as measured by MCDC testing criteria.

4.2 Stability and Coverage Relationships Development – Task 2

Stability in legacy flight control systems is not mathematically guaranteed between design points, but is evaluated in test. These tests specifically look for resonance modes in the air vehicle's structure that could possibly couple with control modes leading to unstable behaviour.

As stability parameters and the mode logic overlap it was suggested that some underlining relationship may exist. It would be desirable to have a direct relationship between coverage parameters and stability parameters of the system. The intention was to try to reduce the apparent redundancy of multiple control loops differing only slightly due to mode requirements. It was hoped that beginning with stability parameters and working backwards into the hierarchy logic, relationships would be uncovered which would lead to an overall reduction in the complexity of the system. Not only would this strategy reduce the complexity of the system, but it would also contribute to bounding the complexity of the system.

Though a strategy to link stability parameters and coverage constraints through the mathematical framework known as Logical Matrix Algebra (LMA) was unsuccessful, pursuing this strategy required the further development of the Logical Matrix Algebraic Network (LMAN). Figure 5 illustrates that a LMAN, like traditional neural networks, is a two-stage matrix transformation.

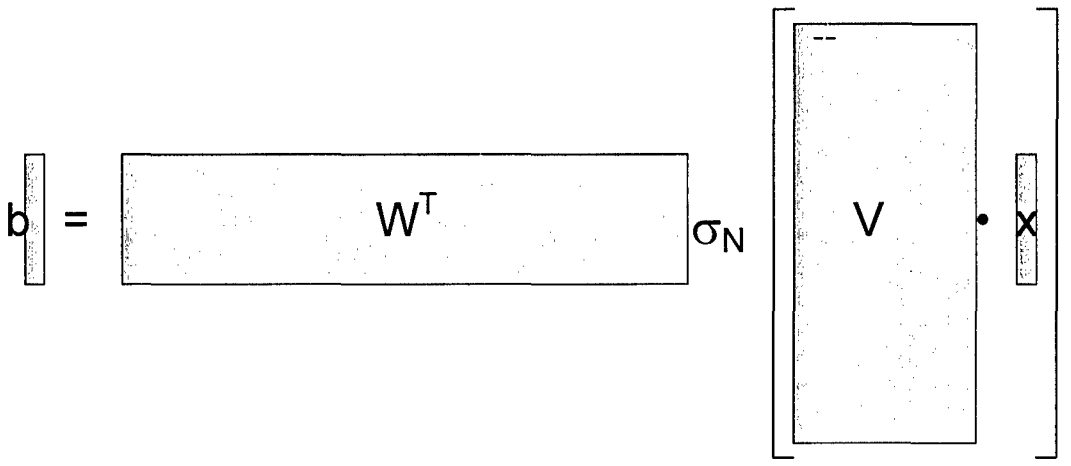


Figure 5. Logical Matrix Algebraic Model Free Estimator

The LMAN's unique learning algorithm, real-time implementation and safety critical properties (including unambiguously defined weights), were applied from work previous to the current research.

The baseline control model representative of basic elements in legacy flight control systems (i.e., a truth model) was analysed using standard techniques (i.e., root-locus, bode etc.). A duplicate system was also constructed deliberately adding an unstable pole. A third system was constructed using a Logical Matrix Algebraic Network (LMAN) model free estimator and shown to be equivalent to the baseline control model. A simulation using the LMAN model free estimator and the unstable baseline model was created using the Simulink tool. The baseline model and its LMA approximation were compared to establish any constraint relationships between them.

4.3 Testable Control System Architectures Development & Analysis – Task 3

4.3.1 Safety Assurance Monitory Development

Equivalent system representations, such as LMAM model free estimators, can be used in a number of ways to increase overall system safety. One important way is to create a monitor enforcing a System Safety Envelope (SSE), Figure 6. Such a Safety Assurance Monitor (SAM) would be one element in a system architecture specifically placed to implement safety-critical requirements. It effectively bounds the output responses of the system. The unstable baseline model was simulated and placed in parallel with its LMAN equivalent. The LMAN equivalent was built form the test vectors of the baseline model and approximated the baseline model until the baseline model approached its unstable region. At this point the LMAN approximation saturated. Since the two models were running in parallel their outputs were compared for

divergence with a default to the LMAN estimator when the divergence exceeded a designated limit.

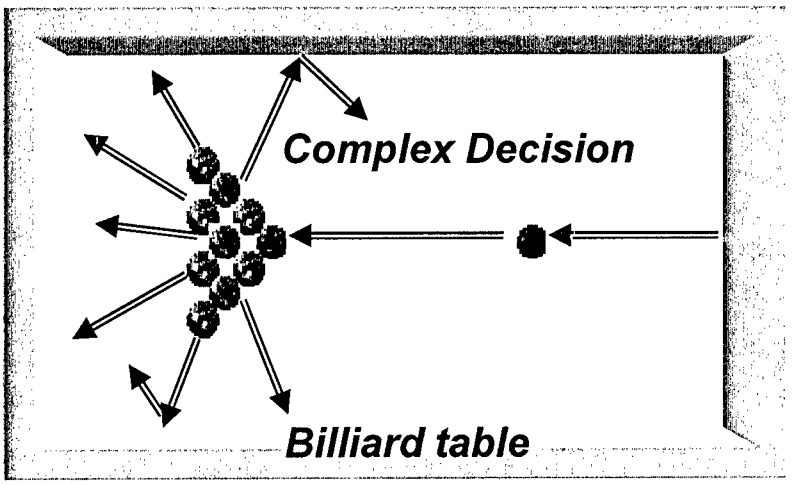


Figure 6. System Safety Envelope enforced by Safety Assurance Monitor (SAM)

This monitoring was applied to demonstrate that an unstable feedback control system could be stabilized. However, the concept can be applied system wide to all variables input, output and observable state variables.

It is noted form legacy programs, that system models contain numerous limit functions or blocks whose purpose is to prevent output signals (primarily actuator signals) from being over driven. These limit functions or model blocks are scattered throughout the system model. Each limit block adds to the system's complexity adding addition constrains and associated tests everywhere they are used demonstrating hidden redundancy throughout the system. It may be argued, that once the limit utility in all its variations is verified, then all instances of the limiting function are verified throughout the model. However, there is no guarantee that every place in the model requiring a limited signal has been identified and the utility function applied. Further,

from a model coverage point of view such limit blocks make large systems significantly more difficult to verify.

A system wide monitor limiting output signals no matter how those signals are generated gives 100% coverage of the model. No gaps or forgotten cases. Each output signal is limited as appropriate without any additional coverage paths being added to the system. The complexity of the system's behaviour is bounded without having to completely analyse the underlining complexity of the system itself.

A unique feature of this LMA based monitor or SAM is that it is built from the test cases of the system it is monitoring. As system development is inherently iterative, not all test vectors of the original system need be known initially. In particular, initial test vectors may be generated directly from the system's requirements. Desired system response can be applied simply by designating the appropriate input and state conditions and defining the required outputs. Not only does LMAN monitor the original system, but it simulates the requirements of the system. This will be seen in the development of Correct-by-Construction Artificial Neural Networks (CCANN)s. These LMANs are verifiably input/output bounded stable.

4.3.2 Correct-by-Construction Artificial Neural Network Development

System wide monitoring via functional estimation is one component of a general Safety-Critical System Architecture. Another component is the estimation of highly non-linear functions as low level elements of such systems such as feedback loops (see figure 7).

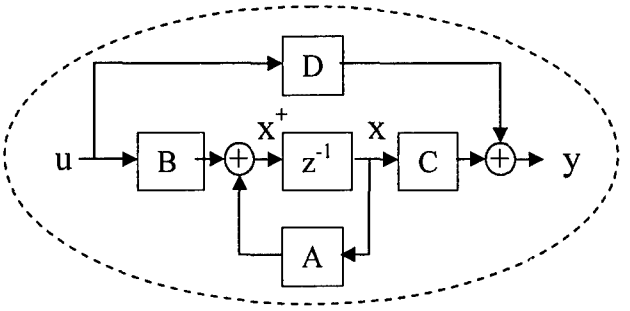


Figure 7. Correct-by-Construction Artificial Neural Network (CCANN)

LMA was used to construct a verifiable neural network non-linear function estimator. A number of system transfer functions were generated using MathWorks Simulink tool. Several input functions were applied to these systems and their corresponding output functions were recorded. From the input and output functions a LMAN was generated correctly estimating the transfer function.

A review of literature dealing with the verification of artificial neural networks was initiated. The general difficulties of verifying such networks were considered and compared with the LMAN function estimator.

4.4 Test Case Reduction Feasibility Assessment – Task 4

Additional strategies for reducing and/or managing system complexity were considered to determine the feasibility and ultimately the effectiveness of employing these strategies in developing safety-critical flight control systems. Since, LMANs are mathematical transformations; some strategies employed in related fields with such transformations were investigated for their appropriateness in reducing test cases/vectors of systems using LMANs.

4.4.1 Additional Strategies for Managing System Complexity

Large systems may be viewed as analogous to integer multiplication. One factor may represent external inputs, the other internal states and their product representing the systems output. Instead of factors composed of a sequence of digits our system factors are vectors of variables. Combinatorial explosion type system complexity may be likened to the factoring of Public Keys used in automated encryption schemes. Key strategies required to address this type of complexity include, parallel processing, methods for quickly analysing failed cases, exploiting the use of partial products (i.e., partial proofs, assertions etc.) and parameter profiling.

Parallel processing in factoring extremely large odd integers is easily done by noting that the least significant digit (LSD) of any product 'N' leads to a small number of possible LSD combinations in the product's factors. For example, let LSD(N) = 1, there are only three possible combinations for the LSDs of its two factors, $\{(1 \times 1), (3 \times 7), (9 \times 9)\}$. Note: the product of all prime numbers (except 2, 5) end in LSDs of 1, 3, 7, or 9, the above three combinations unequally divide up the search space into mutually exclusive and exhaustive sets. Parallel processing does not reduce the size of the search space, though it significantly speeds up the search process. LMA's matrix notation allows for the system description to be conveniently broken up into various parallel subsystems.

It is more efficient to multiply potential factors and comparing their partial product to the known result than it is to divide a factor into the known product and look for a remainder. It is known that the LSDs of two factors uniquely define the LSD of their product. The 2-LSDs of each factor uniquely define the 2-LSDs of their production and so on. By multiplying two potential factors from their LSDs forward their product's value will emerge one digit at a time starting at its LSD. At any point where the calculated product does not match the previously

known result, the test fails and the combinations of factors may be rejected. For systems using algebraic transforms such as LMA and Fourier transforms, consistency checks are also possible. Recall: Fourier transforms input signal characteristics (i.e., symmetry, periodicity etc.) map to the output signal characteristics providing for consistency checks of the calculations.

Probability theory indicates that a unique sequence of numbers becomes less and less likely the longer the sequential string. If two potential integers are multiplied until their calculated product disagrees with the known result and the length of the correct string of digits is noted, this indicates how close the two integers are to the correct solution. If the calculated product yields a correct string of twenty digits, we may assume that it is likely that the initial fifteen or so digits of its two factors are correct. Fixing one of the factors and testing it against potential factors having the same lower string as the second factor enables the scanning of the search table with greater efficiency. Note: the initial string of fifteen or so digits need not be recalculated at each test, it having been previous calculated. Again, this strategy does not reduce the size of the search space, but it explores the search space more efficiently. How to use this strategy with LMA represented systems is currently being investigated.

Creating a profile of integer parameters (i.e., characteristics) leads to a reduction of the size of the search space. In our analogy, Modulo-4 values of any integer may be calculated from its last two digits. Knowing the Modulo-4 value of the product determines the combination of modulo-4 values of its factors. Given the product 'N' is 3(mod4) requires that its factors obey the rule, 1(mod4) × 3(mod4). This effectively reduces the search space as 1(mod4) × 1(mod4) and 3(mod4) × 3(mod4) are not allowed.

The above analogy demonstrates that strategies such as partial products, system parameter profiles etc. are important in managing combinatorial type complexity. These strategies require some kind of relationship between adjacent digits within integers. In actual safety-critical control systems, relationships (i.e., constraints) between system variables are obtained through physical laws (i.e., the law of thermal dynamics and motion etc.) and logical assertions. The LMA system representation has mechanisms for applying both types of system constraints. Our research to date has investigated applying logical assertions and variable boundary constraints. Further investigation in applying physical constraints and system symmetries are the next logical step for future research.

The payoff for using these tools and strategies for managing complexity in verification of safety-critical controls is a simplified, systematic and cost effective (i.e., affordable) way of verifying large highly complex systems.

5. RESULTS / ACCOMPLISHMENTS

5.1 Baseline Control Model Development & Analysis – Task 1

The analysis of the baseline and the other equivalent model representations in Task_1 clearly show room for reduction in systems complexity as determined by the number of test cases. Analysis of legacy safety-critical system development and current model based system development processes indicated that complexity management begins with customer requirements and must be maintained at every phase of developmer.t. System development must consider both the design of functionality and the verification of the system at each phase of system development. That is, 'begin with the end in mind' and keep the end in mind throughout the process. This requires system developers to conscientiously shift their mode of thinking from

functionality to pre/post conditional constraints, toggling their mode of attention as they go. At each stage, the engineer must remember the adage, 'the clearer the picture, the surer the success.' This means both his understanding and communicating that understanding to others throughout the process.

Elements of a complexity space were identified. These elements are interrelated reducing one element may increase or decrease another. Two types of constraints were classified, those that are derived from logical considerations (i.e., human decisions) and those derived from the physical environment. No direct relationship was found for these constraints though they use the same variables.

5.2 Stability & Coverage Relationships Development – Task 2

The LMAN was used as a monitor for the unstable system and demonstrated (1) the baseline system could approach an unstable region, (2) the LMAN model free estimator was able to control the stability of the baseline system by over riding the systems output in the unstable region. Importantly, the LMAN model prevented the instability by bounding the output.

In spite of significant effort, a direct relationship between the stability parameters of the system and its coverage constrains was not achieved. However, by applying the formal language techniques and particularly Logical Matrix Algebra (LMA) we were able to exploit the common logical and functional constraints of the system to develop specific tools. First the Safety Assurance Monitor (SAM) a system architectural stability control tool. Second an alternative neural network, Correct-by-Construction Artificial Neural Networks (CCANNs) with verifiable properties vital for use in safety-critical systems.

These tools address the original intent of Task_2 which was to reduce system complexity by exploiting common stability and conditional constraints.

The development of SAM and CCANN are the primary contributions of this research. They are available for immediate and effective contributions to the design and development of safety-critical flight control systems. Their development represents a new point of view reflecting less on the dynamics of calculus and more on the biology of nature.

5.3 Testable Control System Architectures Development & Analysis – Task 3

At some point, basic research needs to become a practical application. Two Safety-Critical System Architectural components were developed. 1) Safety Assurance Monitor (SAM) for use as a system back up enforcing a system safety envelope illustrated in Figure 6. 2) Correct-by-Construction Artificial Neural Network (CCANN) which addresses verification issues for neural networks used in safety-critical situations illustrated in Figure 7.

CCANNs directly address the management of test complexity by reducing some of the longest running tests and training times inherent in verifying traditional neural networks. Low cost safety-critical verification of non-linear function estimators will become vital as these artificial intelligent system components find their way in to advanced flight controls systems. Both computational complexity and semantic ambiguity are reduced, thus, enhancing safety assurance in both the development process and the final product.

The architecture of safety-critical control systems is specifically addressed in how SAM's are used to apply the systems safety envelope. While at the individual function level CCANN's allow for verifiable non-linear function estimation. Dynamic system learning both prior to flight operations and real time during flight operations have been considered and future research will seek to flesh out appropriate applications in this area. Real time verification of the LMAN's dynamic learning process was also considered and again future research will follow through on these issues. See the New Discoveries section.

5.4 Test Case Reduction Feasibility Assessment – Task 4

The feasibility of reducing the number of test cases of existing safety-critical flight control systems was assessed. That there is room for reduction has been demonstrated. Whether or not those reductions can be made cost effectively depends upon the adoption of a number of methods, tools and the above newly developed techniques. Prevention of complexity growth has the greatest immediate potential for complexity management and requires only the application of existing tools and known best practices. Specific use of safety design techniques/tools such as SAMs or CCANNs will depend upon the application. A number of strategies for managing specific types of complexity have been addressed throughout this report. Safety-Critical Flight Control Systems are currently so complex that the use of all of these techniques will be necessary to adequately manage these systems in the future. New techniques will become available as biological systems are studies and nature's divine design is copied. A combination of approaches, techniques, tools, processes and practices will be necessary to adequately manage current and future safety-critical flight control systems. The some of the most important issues in managing complexity are not technical, but human in origin. Human ability to adapt new approaches and tools will determine the success of any new developments. The following addresses some barriers to appropriate implementing of the resources available.

6. NEW DISCOVERIES

6.1 Safety Assurance Monitor – SAM

LMA forms the mathematical foundation for both the Safety Assurance Monitor (SAM) [11, 12, 13, 14, 15] and Correct-by-Construction Artificial Neural Network (CCANN) [8, 9, 10]

which are two applications of the same technology. For an introduction to LMA see the appendix. [1, 3, 5, 6, 7, 8]

6.2 Correct by Construction Artificial Neural Networks – CCANN

6.2.1 Introduction

Traditional artificial neural networks (TANN) are constructed as layers of bounded, weighted sums. Learning uses optimization algorithms adjusting weights to yield appropriate responses to a given set of training data. Adaptive neural networks go further, adding online updating algorithms in what amounts to continuous learning. Both the initial and online learning algorithms manipulate data in such a way, that most of the concrete meaningful information is obscured. Simply put, information is hidden by the math. Correct by Construction Artificial Neural Networks (CCANN) keeps information visible by taking an alternative path to the same solution. Some of the differences between TANNs and their CCANN counter parts simply reflect a reordering of the sequence of operations. There are, however, significant differences between them as well. CCANNs do not use optimization algorithms, that is, they do not rely on differential equations as a learning mechanism. CCANNs do not learn by updating their weight values. Instead, CCANNs learn by adding nodes keeping all previous weights fixed or deleting them as redundancies. At first, this approach seems to be unfeasible, however, upon a little investigation, such an approach becomes practical. Importantly, information is kept "visible" turning what was a semantic black-box into a white-box. This approach eliminates much of the verification difficulty arising from the uncertainties of black-box considerations. CCANNs can be made adaptive in a straight forward extension of their static model. Using CCANNs as part of a multiplex software system "avoids" simultaneous failure because their development approach

is significantly different than traditional models. CCANNs have a number of advantages over their TANN counter parts both in the ease of their construction and verification. This makes their inclusion into existing development programs low cost and low risk.

There are two types of Correct by Construction Artificial Neural Networks, (1) those built from input/output data, and (2) those built from a rule base.

A key assumption in creating neural networks with more visible information is that there is nothing sacred about the order of mathematical operations in building functional estimators. For example, it is a convention to use Sum-of-Products for matrix multiplication. However, it has been shown that substituting Product-of-Sums in matrix multiplication resolves conflict in modelling flexible manufacturing systems [3]. In our current application allowing for non-standard ordering of algebraic operations gives additional freedom in maintaining informational visibility throughout the calculations. During the development of Correct by Construction Artificial Neural Networks – CCANNs the author will feel free to break with convention whenever it is necessary to maintain the visibility of information.

Correct-by-Construction Artificial Neural Networks are built upon the mathematical formalism of Logical Matrix Algebra (LMA). The reader is referred to the appendix below for an introduction to LMA.

6.2.2 Data Constrained - Correct by Construction Artificial Neural Networks

The goal will be to construct a model free estimator (i.e., functional approximation). Given an input/output training data set, construct an estimator that minimally represents the functional mapping of data pairs.

Let all possible input points (i.e., domain) be denoted as $X = \{x_1, x_2, \ldots, x_n\}$. Any training inputs will be a subset of the input domain $X_t \subseteq X$.

Let all possible output points (i.e., range) be denoted as $Y = \{y_1, y_2, \ldots, y_m\}$. The associated outputs of training data inputs will be a subset of the range $Y_t \subseteq Y$. Training data form pair wise mappings $(x_i, y_i) \in X_t \times Y_t$.

The general form of a two-stage algebraic transformation may be given as,

$$y = W^T \circ (V \bullet x) \qquad \text{or} \qquad y = g(W, f(V, x))$$

where the '•' (i.e., dot) and '∘' (i.e., circle) operations will be defined as required.

The following "outline" shows our development strategy. Working from the result (i.e., mapped I/O data pairs), and redefining matrix multiplication as necessary,

Given the mapping $y = Ax$ where x, y are column vectors, A is the system matrix.

$y = (yx^{-1})x$ substituting matrix A with $A = yx^{-1}$.

$y = y(x^{-1}x)$ by regrouping.

$y = y(x^{-1} \bullet x)$ define '•' such that, $e = (x_i^{-1} \bullet x_i) = 1$, else $(x_j^{-1} \bullet x_k) = 0$.

$y = y \circ e$. define '∘'.

$y = y$.

This "outline" is not a proof, but a sketch of a strategy. Note: vector x^{-1} in general, does not exist in conventional algebra. Following this sketch,

$y = Ax$ as above.

Substitute the single-stage transformation system matrix 'A' with, a two-stage transformation $A = W^T V$. Notice, that matrices W and V can be any length and must follow sizing rules of linear algebra.

Let the columns of W^T be "paired" with the rows of V (this is a key concept). Let each column of W^T represent individual output vectors y_i and each row of V represent their associated input vectors x_i. Thus, for the present, $W = Y_t$ and $V = X_t$ where Y_t and X_t have the same ordering. This arrangement forces deterministic mapping, each input is paired with its associated output. Hence, the meaning of the weights in W^T and V are known (i.e., visible). They are the actual input/output data values.

$y = (W^T V)x.$ substituting the system matrix 'A'

$y = W^T(V \bullet x).$ Regroup and define the 'dot' operator.

In general the 'dot' operator $V \bullet x$ is generic, it may represent any appropriate linear or nonlinear closed form function $e = f(V, x)$. The first stage operation isolates (i.e., identifies) all input points having a similar property in the input domain. The intermediate output maps this property 'onto' the output range.

The present goal is to construct a partial result 'e' that evaluates to the value '1' when there is an appropriate match (i.e., $x = v_i$). This is equivalent to multiplying the input 'x' by its inverse 'x^{-1}.' For data constraint construction, a first approximation to $e = f(V, x)$ will be,

$$e_i^C = \|x - v_i\|^2 / (x^2 + v_i^2)$$

The numerator is the square of the distance between the current input vector 'x' and any previously recorded input vector 'v_i.' Thus, the partial result vector 'e^C,' represents an ordered listing of the distances of all previously known inputs to the current input. The denominator normalizes this distance measure to within the range of [0.. 2] via the Pythagorean inequality.

We desire that $e_i = 1$, for $x = v_i$. The above function does not do this. To obtain the desired result we simply take its complement, thus

$$e_i = 1 - \|x - v_i\|^2 / (x^2 + v_i^2)$$ where, Complement/NOT: $x^C = 1\text{-}x$

The range of the partial result, 'e' now becomes [-1..1]. Applying the equivalence expression,

$\|x\text{-}v_i\|^2 = x^2 - 2xv_i + v_i^2$, the above formula simplifies to,

$$e_i = 2xv_i / (x^2 + v_i^2)$$

Some key observations are,

$$e_i = \begin{Bmatrix} 1, & x = v_i \\ \approx 1, & x \approx v_i \\ 0, & x \perp v_i \end{Bmatrix}$$ $\approx$ meaning 'nearly equal to'

The final step in determining the first stage result is to filter the partial result vector 'e,'

If $\max(e_i) = 1$, then set all $e_j = 0$, where $i \neq j$.

If $\max(e_i) \neq 1$, then 'select' two values e_i, e_j nearest to '1' and set all other $e_k = 0$. $k \neq i, j$.

This will result in a new 'e' vector padded with zeros. In traditional neural network terms we have selected a 'winning' node(s). In practical terms, we have avoided difficulties in the second stage calculation by eliminating extraneous data. This maintains the visibility of the value's (i.e., 'e') meaning throughout the calculation. Only the best information is retained.

The 'circle' operator is also generic. Here we calculate our output estimate y_{est} as,

$y_{est} = y \circ e.$ defining, the 'circle' operator ,

$$y_{est} = (y_i e_i + y_j e_j)/(e_i + e_j)$$

The weighted sum of the outputs divided by the sum of the weights (i.e., centroid).

The above calculations are based upon two assumptions, that both the input domain and output range are each continuous. Hence, sets (i.e., patches) in the input domain map into sets (i.e., patches) in the output range. xi and x_j identify two points in the input domain nearest to the current input all of which are contained within a continuous set. This set maps onto a continuous range that contains outputs y_i and y_j.

6.2.3 Learning Strategy

The learning strategy is simply to add a new column (i.e., w^T_k) and its paired row (i.e., v_k) for any unknown mapping outside a specified error limit (i.e., error = $|y_{actual} - y_{est}|$). If the current error is within the specified error limit, then the current I/O pair or one of the associated I/O pairs used in calculating the output estimate (i.e., y_{est}) may be eliminated as redundant information.

This learning strategy lends itself to 'point selection' optimization. That is, only those I/O pairs that minimally represent the function are retained. This means that the size of matrices W and V are kept relatively small. Further, this learning strategy (1) avoids issues of training data ordering, (2) is a direct calculation (i.e., does not require convergence), hence, is less time consuming, thus appropriate for real-time, online applications, and (3) is particularly valuable in adaptive networks as new information can directly update the network.

Learning in Correct by Construction Artificial Neural Networks – CCANNs is not based upon differential equations, hence there are no issues of gradients, wells, equilibrium points, limit cycles, local minimums, global minimums, stability issues or step size convergence issues

etc. CCANNs use a direct mapping scheme. They are based upon very few assumptions (i.e., continuity in both the domain and range).

Far from being a disadvantage, differences between direct mapping and modelling based on differential equations lends itself to significant advantages. The advantages are especially apparent when those differences are played off each other. It has been shown that it is not feasible to verify software with mean time between failures (MTBF) rates of 10^{-9}, even for multiplex systems. This is because software models, even when developed by independent groups, will have their underling approach (i.e., the math) in common. This leads to the real possibility of simultaneous failure of both models. Direct mapping is a significantly different modelling approach reaching down to basic assumptions. Thus, it lends itself less susceptible to simultaneous failure when used as an element in multiplex systems or as a monitor. Direct mapping as implemented in a CCANN like other networks tend to fail gradually. Thus, adding additional safety when coupled with traditional models that may fail catastrophically.

There is a price to be paid for direct mapping. That price is the usual trade off between computation and memory. However, CCANNs do not require significantly more memory than TANNs as memory is optimized. Further, with increased speed and lower cost of memory, any additional costs in this direction will be minimized. An additional advantage comes from exchanging of computations (i.e., software) for memory (i.e., hardware). Direct mapping exchanges difficult to verify software for the more readily verifiable hardware. (issue of EMI corrupting of memory values and connections still exist.)

6.2.3.1 Real-Time Calculation issues

Dividing by zero must be avoided. This can easily be done by the simple rule, $v_i \neq 0$.

Dividing the square of the distance by the sum of the squares (i.e., $||x - v||^2/(x^2 + v^2)$) skews the distance values but does not reorder them near the value '1'. The filter removes (i.e., replaces with zeros) all element values of vector 'e' not closest to the value '1.'

$e = 0$ identifies the orthogonally of x and v_i ($x \perp v_i$). This may be exploited (TBD).

CCANNs computations may be broken up into any number of parallel arrangements. This is because the columns of W and the rows of V are paired. At lease three possibilities are, (a) paired blocks of W and V could be assigned to separate processors, (b) individual elements of the output vector 'y' could be calculated independently of the others, (c) input vectors could be grouped together similar to rule based calculations (see logically constrained CCANNs). Each of these methods leads to rapid (i.e., real-time) calculation of very large systems. Therefore, what at first appeared to be unfeasible easily becomes practical.

Online Adaptive Neural Networks are required to operate in real-time. The time it takes to run these optimization programs may become unacceptable. CCANNs run, learn & adapt fast.

6.2.4 Adaptive extension strategy:

Our intent is not to fully develop Adaptive CCANNs, but to suggest an adaptive scheme for CCANNs. A number of schemes may be used to update static CCANNs making them adaptive. One simple scheme would be to tag the input vectors with a timing ID. When evaluating which input vectors are closest to the current input, the most resent known inputs vectors would be used, defaulting to older ones if necessary.

6.3 Logically Constrained - Correct by Construction Artificial Neural Networks

The goal will be to construct a model free estimator (i.e., functional approximation) from a rule base. Given a rule base, construct an estimator that minimally represents a functional mapping. The general form of a two-stage transformation may be given as,

$$y = W^{T} \circ (V \bullet x) \qquad \text{or} \qquad y = g(W, f(V, x))$$

where the '•' (i.e., dot) and '∘' (i.e., circle) operations will be defined as required.

6.3.1 Background to build a rule structure.

Logical Matrix Algebra (LMA) was developed form a mathematical description of Discrete Event Systems [3]. The definitions of LMA are as follows,

Unary Operators: where $r \in R$

NOT (i.e., Complement) $r^C = 1 - r$. Note: $r^C + r = 1$ is similar to $r^C \rightarrow r = 1$

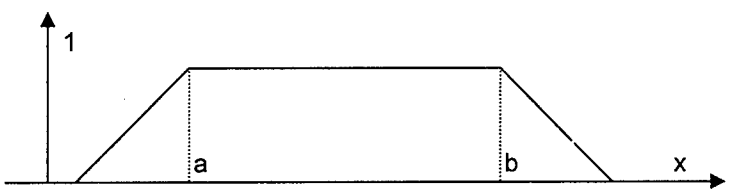
Saturation $\sigma(r) = \min[1, \max(0, r)] = \max[0, \min(1, r)]$

Lemma $\sigma(\sigma(r_1)\sigma(r_2)) = \sigma(r_1)\sigma(r_2)$

Binary Operators: where a, b ϵ R

AND	$\sigma(ab)$	bounded product
OR	$\sigma(a + b)$	bounded sum
Implementation	$a \rightarrow b = \sigma(a^C + b)$	
Equivalence	$a \equiv b = (a \rightarrow b)(b \rightarrow a) = \sigma(a^C + b)\sigma(a + b^C)$	

LMA Rule $\sigma(a^C + x)\sigma(b + x^C)$ where 'a' is the lower and 'b' is the upper bound.



The shape of the LMA rule is an envelope on a continuous variable. It behaves much like paired unit-step functions in interval logic. If sampled, it looks like a fuzzy set. Note: Fuzzy systems quantize and normalize their inputs, usually applying a bit vector.

Example: The fuzzy set 'short' over the variable 'height' is $x_{higth\text{-}short}$ = [1 1 1 1 1 0.5 0 0 0 0 0], each element place (i.e., fit value) represents a specific height (i.e., [1ft, 2ft, 3ft, ..., 10ft]) An input bit vector x = [0 0 0 0 1 0 0 0 0 0] represents an object of height 5-feet. Clearly, this information could be represented as a scalar (i.e., 5). Fuzzy systems in general allow for fuzzy sets as inputs, however, much of the time simple bit vectors are used.

LMA Rules use scalar inputs, but act much like fuzzy sets (i.e., weighting rules).

An LMA rule may be implemented in matrix format via element wise multiplication. Let the input be a vector $x = [x_1, x_2, \ldots, x_n]$. Each element of the partial result vector 'e' represents the intersection of rules.

$e_i = \text{LMARule}_i(x_1) .\times \text{LMARule}_i(x_2) .\times \ldots .\times \text{LMARule}_i(x_n)$, denotes

If $[(l_{x1i} \le x_1 \le u_{x1i})$ AND $(l_{x2i} \le x_2 \le u_{x2i})$ AND ... AND $(l_{xni} \le x_n \le u_{xni})]$, Then e_i.

Note: these are fuzzy like rules in that they have roll-offs around their limits. The slopes of these roll-offs can be specified, but not in this example. They form patches (i.e., cells) in the input space. The consistency and completeness of domain coverage can easily be determined. The intermediate or first stage calculation becomes,

$$e_i = [\sigma(l_{x1i}{}^C + x_1)\sigma(u_{x1i} + x_1{}^C)][\sigma(l_{x2i}{}^C + x_2)\sigma(u_{x2i} + x_2{}^C)] \ldots [\sigma(l_{xni}{}^C + x_n)\sigma(u_{xni} + x_n{}^C)]$$

reordering by grouping lower and upper bounds.

$$e_i = [\sigma(l_{x1i}{}^C + x_1)\sigma(l_{x2i}{}^C + x_2) \ldots \sigma(l_{xni}{}^C + x_n)] .\times [\sigma(u_{x1i} + x_1{}^C)\sigma(u_{x2i} + x_2{}^C) \ldots \sigma(u_{xni} + x_n{}^C)]$$

$e = L^C \bullet x .\times U \bullet x^C$. where '•' is a Product-of-Sums multiplication

therefore

$y = W^T \circ (L^C \bullet x .\times U \bullet x^C)$ recall, $.\times$ is element-wise multiplication

where the columns of W^T are associated with elements of 'e' that represent logical inferences. Like fuzzy systems the '∘' (i.e., circle) operator, here performs a centroid calculation. However, unlike fuzzy systems, the input vector does not need to be a fuzzy set or bit vector, here they are an array of scalar elements.

6.3.2 Learning

Unlike data constrained CCANNs, logically constrained CCANNs rely on an inference bank. Learning therefore is usually prior to implementation. However, adaptive updating algorithms may be added.

6.3.3 Adaptive Logically Constrained CCANNs

An adaptive update scheme for a logically constrained CCANN could update the rule bank. Adding timing markers to the lower and upper limits would allow the age of the information to be used in updating. Another method would be to follow the IFCS – dynamic cell update strategy [4]. Again, the purpose of this paper is not to completely develop adaptive networks, rather to suggest that adaptive CCANNs are possible.

7. ACKNOWLEDGEMENTS

This work was sponsored by the Air Force Office of Scientific Research, USAF, under contract number F49620-03-C-0025 and conducted at Lockheed Martin Aeronautics, Fort Worth, Texas. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the U.S. Government or Lockheed Martin Aeronautics.

8. REFERENCES

[1] Boren, "Introduction and Development of Logical Matrix Algebra", *23rd International Holiday Mathematics Symposium*, 1999.

[2] Blackburn, and Busser, T-VEC Modeling Language, T-VEC Technologies, Inc., 1996

[3] Lewis, Pastravanu, and Huang, "Controller design and conflict resolution for discrete event manufacturing systems," Proc. IEEE Conf. Decision and Control, pp. 3288-3294, San Antonio, Dec. 1993.

[4] Clough, "Autonomous UAV Control System Safety – What Should It Be, How Do We Reach It, And What Should We Call It?," 2000 NAECON.

[5] Garnier and Taylor, Discrete Mathematics for New Technology; ed. 2, Adam Higer Pub.

[6] Boren, Lewis, and Tacconi,, "Nonlinear Network Model for Discrete Event Systems," UTA research report, 1997.

[7] Golub and Van Loan, Matrix Computations, 2nd ed., Baltimore, MD: Johns Hopkins University Press, 1989.

[8] Kosko, Bart, "Neural Networks and Fuzzy Systems," Prentice-Hall, 1992.

[9] Silva and Krogh, "Formal verification of hybrid systems using CheckMate: a case study," *2000 American Control Conference*, Chicago, June 2000.

[10] Haykin, Neural Networks, A Comprehensive Foundation, Macmillan, 1994.

[11] Leonessa, Haddad, and Chellaboina, "Nonlinear Robust Hierarchical Control for Nonlinear Uncertain Systems," *J. Math. Prob. Engin.*, vol. 5, pp. 499-542, 2000.

[12] Kokotovic and Arcak, "Constructive Nonlinear Control: Progress in the 90's," Automatica, Vol. 37, No. 5, May 2001, pp. 637-662.

[13] Goncalves, Megretski, and Dahleh, "Global Stability of Relay Feedback Systems," IEEE Transactions on Automatic Control, 2001.

[14] Sontag, Arcak, and Angeli, "A unifying integral ISS framework for stability of nonlinear cascades," submitted.

[15] Teel, Moreau, and Nesic, "A unified framework for input-to-state stability in systems with two time scales," *IEEE Transactions on Automatic Control*, 2001.

9. PERSONNEL SUPPORTED DURING DURATION OF GRANT

Dale W. Boren, P.E. Embedded Software Engineer Senior Staff,
LM Aeronautics Co- Fort Worth, Texas

James M. Buffington, PhD Research Engineering Manager
LM Aeronautics Co- Fort Worth, Texas

10. HONOURS AND AWARDS – N/A

11. AFRL POINT OF CONTACT

Lt Col. Sharon Heise, USAF, PhD Acting Director, Mathematics & Space Sciences,
Program Manager, Dynamics & Control,
Air Force Office of Scientific Research
875 Randolph Street, Ste. 324, Rm 3112
Arlington, VA 22203

12. INTERACTIONS – N/A

13. TRANSACTIONS

This fundamental test reduction/complexity management concept will be further matured in the AFRL VVIACS and CertaFCS applied research programs by LM Aero. Point of contact at LM Aero is James Buffington, PO Box 748, Mail Zone 9338, Fort Worth, TX, 76101, Phone: 817-935-1030.

14. APPENDIX

14.1 Development of Logical Matrix Algebra

Logical Matrix Algebra (LMA) is an algebraic structure defined as an Abelian Monoid-Ring with Unit. LMA extends logical functionality over number sets beyond the binary. Simply, LMA constructs the logical operators 'AND' and 'OR' from the more general arithmetic operations of multiplication and addition with the aid of a unitary limiting operator (i.e., saturation function). 'NOT' or Complement is by its nature defined over the real number set.

(a AND b) is defined as $\sigma(a*b)$ $\quad a, b \in R$ (Def.1)

where the saturation function, $\sigma(\_)$ limits the product of 'a' and 'b' to the range [0, 1].

(a OR b) is defined as $\sigma(a+b)$ $\quad a, b \in R$ (Def.2)

where the saturation function, $\sigma(\_)$ limits the sum of 'a' and 'b' to the range [0, 1].

The saturation function may be defined in a number of ways. For our purposes we will define the saturation function as,

$\sigma(r) = \min(1, \max(0, r))$ or $\sigma(r) = \max(0, \min(1, r))$ $\quad r \in R$ (Def.3)

These definitions are carefully extended from scalar to matrix notation. Hence, truth-tables and like logical functions can be constructed using LMA notation. LMA freely mixes both arithmetic and logical equivalent operations. To the above simple construction of a new algebra is added one more significant feature. LMA does not limit the use of its additive and multiplicative operators (i.e., OR, AND) to any specific order when using matrix notation. We may define a '•', denoted 'dot', vector operator as normal matrix multiplication having addition

and multiplication exchanged for each other. That is, we substitute the usual multiplication for addition, and the usual addition for multiplication.

$$a^T \bullet b \equiv \begin{bmatrix} a_1 & a_2 & \dots & a_q \end{bmatrix} \bullet \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{bmatrix} = \sigma(a_1 + b_1)\sigma(a_2 + b_2)\cdots\sigma(a_q + b_q) \qquad \text{(Def.4)}$$

Applying the 'dot' operator to two matrices.

$$A^T \bullet B \equiv \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} \sigma(a_{11} + b_{11})\sigma(a_{12} + b_{21}) & \sigma(a_{11} + b_{12})\sigma(a_{12} + b_{22}) \\ \sigma(a_{21} + b_{11})\sigma(a_{22} + b_{21}) & \sigma(a_{21} + b_{12})\sigma(a_{22} + b_{22}) \\ \sigma(a_{31} + b_{11})\sigma(a_{32} + b_{21}) & \sigma(a_{31} + b_{12})\sigma(a_{32} + b_{22}) \end{bmatrix}$$

Along with the appropriate Lemmas, Theorems and Proofs [2], LMA provides a powerful tool for analysing hybrid dynamical systems. Let us begin with a simple static logic problem.

Example 1: OR Operator

Given $\mathbf{u} = [u_1, u_2]^T$ as a binary input vector. Also, **Y** and **U** as weighted matrices of appropriate dimensions, then

$$y = \sigma\left(Y^T \times \left(U^C \bullet u\right)\right) \qquad \text{the logical operation } (u_1 \text{ OR } u_2)$$

Expanded into matrix notation

$$y = \sigma\left(\begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \times \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \right) \qquad \text{(E.1.0)}$$

The weights of the matrices **Y** and **U** are not arbitrary. They are take directly from the truth table definition of the 'OR' operation. By applying the 'dot' operator

$$y = \sigma\left(\begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{pmatrix} \sigma(1+u_1)\sigma(1+u_2) \\ \sigma(1+u_1)\sigma(0+u_2) \\ \sigma(0+u_1)\sigma(1+u_2) \\ \sigma(0+u_1)\sigma(0+u_2) \end{pmatrix} \right) \qquad \text{(E.1.1)}$$

$$y = \sigma(1 \times \sigma(1+u_1)\sigma(0+u_2) + 1 \times \sigma(0+u_1)\sigma(1+u_2) + 1 \times \sigma(0+u_1)\sigma(0+u_2)) \qquad \text{(E.1.2)}$$

Reduced by $\sigma(1) = 1$ and by $u_1, u_2 \in$ Binary

$$y = \sigma(u_1 + u_2 + u_1 u_2) \qquad u_1 + u_2 \supseteq u_1 u_2 \text{. by truth table} \qquad \text{(E.1.3)}$$

$$y = \sigma(u_1 + u_2) \qquad \text{(E.1.4)}$$

Therefore $u_1 OR u_2 \equiv \sigma(u_1 + u_2)$ for $u_1, u_2 \in$ Binary (E.1.5)

Note: The Exclusive OR function can also be obtained in a similar manner. The look and feel of linear algebra remains, but nonlinear operations may be performed.

The following relationships will be needed in the upcoming examples.

The definition of a Complement or 'Not' operator

scalar: $a^C = 1 - a$ the standard definition (Def.5)

vector: $\mathbf{a}^C = [1] - \mathbf{a}$ where [1] denotes a vector with all elements equal to '1'.

matrix: $\mathbf{A}^C = [1] - \mathbf{A}$ where [1] denotes a matrix with all elements equal to '1'.

From this we conclude (at the scalar level) the following,

$a^C + a = 1$ where '1' is the multiplicative identity

$a \rightarrow b = a^C$ OR b is a standard logical inference identity

a^C OR $b = \sigma(a^C + b)$ from Definition 2 above

Also let $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_q]$ and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_q]$

where $\mathbf{a}_k$ and $\mathbf{b}_k$ are p-by-1 vectors and **A** and **B** are p-by-q matrices.

These conclusions come together at the vector level in the following key observation. The row-column vector product using the 'dot' operation provides a key association between two vectors.

$\mathbf{a}^{CT}\bullet\mathbf{b}$ = is a scalar, and if $\mathbf{b} = \mathbf{a}$, then

$$a^{CT} \bullet a = \begin{bmatrix} a_1^C & a_2^C & \cdots & a_q^C \end{bmatrix} \bullet \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_q \end{bmatrix} = \sigma\left(a_1^C + a_1\right)\sigma\left(a_2^C + a_2\right)\cdots\sigma\left(a_q^C + a_q\right) = 1$$

This vector product uniquely identifies a matching of two vectors (i.e., $\mathbf{a}^{CT}\bullet\mathbf{a} = 1$). Further, if $\mathbf{a}^{CT}\bullet\mathbf{b} \leq 1$, then **b** is a term wise subset of **a**. If **b** is a term wise superset of **a**, the saturation function will cause the product to yield a value of '1' [15].

Any matrix-vector product forms a column vector with each element representing the scalar product of two vectors. The first vector is an individual row of the matrix, and the second vector is the input vector itself.

$$A \bullet x = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \sigma\left(a_{11} + x_1\right)\sigma\left(a_{12} + x_2\right) \\ \sigma\left(a_{21} + x_1\right)\sigma\left(a_{22} + x_2\right) \\ \sigma\left(a_{31} + x_1\right)\sigma\left(a_{32} + x_2\right) \end{bmatrix}$$

If any row of **A** matches the complement of the input vector (i.e., $\mathbf{x}^C$), then the output element corresponding to that row will have the value of '1', else the output will lie in the range [0, 1]. This matrix-product can be said to form a 'generalized logical inference'. It is directly related to fuzzification of Fuzzy Systems and the Enabling vector of Discrete Event Systems (DES) [6]. Without further explanation, it can be shown that,

$(\mathbf{A}^{TC} \bullet \mathbf{A})$.* $(\mathbf{A}^{T} \bullet \mathbf{A}^{C}) = \mathbf{I}$ where '.*' denotes term wise multiplication

Example 2: Static Mapping (i.e., pattern-association problem) [ref. 7, page 617]

The procedure used here is based on the classical method of least squares.

Given a set of input vectors denoted by $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_q$ and a corresponding set of desired response vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_q$, find a mapping or matrix operation that minimizes the difference between the desired response $\mathbf{b}_k$ and the actual response $\mathbf{M}\mathbf{a}_k$ for k = 1, 2, ..., q.

In linear algebra such a mapping may be denoted as $\mathbf{b} = \tilde{\mathbf{M}}\mathbf{a}$ where $\tilde{\mathbf{M}}$ is given by

$\tilde{\mathbf{M}} = \mathbf{B}\mathbf{A}^+$ $\mathbf{A}^+$ is a pseudo inverse of the matrix **A**. (E.2.3)

The goal being to derive a matrix $\tilde{\mathbf{M}} = \mathbf{B}\mathbf{A}^+$, such that when it is multiplied by **A** we obtain

$$\tilde{\mathbf{M}}\mathbf{A} = \mathbf{B}\mathbf{A}^+\mathbf{A} \quad \text{(E.2.4)}$$

If $\mathbf{A}^+\mathbf{A} = \mathbf{I}$ where **I** is the identity matrix, then

$\tilde{\mathbf{M}}\mathbf{A} = \mathbf{B}$ the desired output. (E.2.5)

For linear algebra the pseudo inverse matrix $\mathbf{A}^+$ is defined by (7, 8) as,

$$\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T. \quad \text{(E.2.6)}$$

The realization of this pseudo inverse depends upon the linear independence of the set of input vectors used in constructing matrix **A**. If **A** is not full rank, then the inverse of $\mathbf{A}^T\mathbf{A}$ cannot be calculated.

Logical Matrix Algebra provides a nonlinear solution to the mapping problem that is not dependent upon the rank of **A**. This solution, however, comes at a price. The pseudo inverse $\mathbf{A}^+$ in LMA notation cannot be isolated. However the constraint $\mathbf{A}^+\mathbf{A} = \mathbf{I}$ can be represented as,

$$(\mathbf{A}^{CT} \bullet \mathbf{A}).*(\mathbf{A}^T \bullet \mathbf{A}^C) = \mathbf{I} \quad \text{(E.2.12)}$$

The mapping matrix operation becomes $\mathbf{B} = \mathbf{B}(\mathbf{A}^{CT} \bullet \mathbf{A}).*(\mathbf{A}^T \bullet \mathbf{A}^C)$ (E.2.13)

Operationally, $\mathbf{b}_k = \tilde{\mathbf{M}}\mathbf{a}_k$, becomes $\mathbf{b}_k = \mathbf{B}(\mathbf{A}^{CT} \bullet \mathbf{a}_k).*(\mathbf{A}^T \bullet \mathbf{a}_k^C)$ (E.2.14)

Though the notation may not be as compact, the LMA solution to the mapping problem is functionally superior. There is no limitation on the rank of **A**. No division is performed in calculating the mapping. Hence, there is no danger of a divide by zero. This procedure requires no more memory than the least squares procedure. Finally, LMA allows for nonlinear mappings to occur including mappings for inputs of zero.

14.1.1 Properties of Logical Matrix Algebra

Logical Matrix Algebra (LMA)

Bivariable Operators

AND: $\sigma(ab)$ bounded prod

OR: $\sigma(a + b)$ bounded sum

Univariable Operators

NOT: $r^C = 1 - r$ over all reals

Saturation:

$\sigma(r) = \min[1, \max(0, r)] = \max[0, \min(1, r)]$

Commutative Laws:

$\sigma(ab) = \sigma(ba)$

$\sigma(a + b) = \sigma(b + a)$

Absorption Laws:

$\sigma(a\sigma(b + c)) = \sigma(a)$

$\sigma(a + \sigma(bc)) = \sigma(a)$

Identity Laws:

$\sigma(a + 1) = 1$

$\sigma(a + 0) = \sigma(a)$

$\sigma(a \times 1) = \sigma(a)$

$\sigma(a \times 0) = 0$

Involution Law:

$(a^C)^C = a$

Inference:

$a \rightarrow b = \sigma(a^C + b)$

Equivalence:

$a \equiv b = \sigma(a^C + b)\sigma(a + b^C)$

Distributive Laws:

$\sigma(a\sigma(b + c)) = \sigma(\sigma(ab) + \sigma(ac))$

$\sigma(a + \sigma(bc)) = \sigma(\sigma(a + b)\sigma(a + c))$

De Morgan's Law:

$\sigma(a + b)^C = \sigma(a^C b^C)$

$\sigma(ab)^C = \sigma(a^C + b^C)$

Associative Laws:

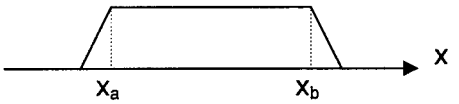
$\sigma(a\sigma(bc)) = \sigma(c\sigma(ab))$

$\sigma(a + \sigma(b + c)) = \sigma(c + \sigma(a + b))$

Complement Laws:

$a \rightarrow a = \sigma(a^C + a) = 1$, for all reals

$\sigma(a^C a) = [a^C a$, for $0 < a < 1$; else $0]$



LMA Rule: $\sigma(x_a{}^C + x)\sigma(x_b + x^C)$

Lemma 1 $\sigma(\sigma(r_1)\sigma(r_2)) = \sigma(r_1)\sigma(r_2)$

Lemma 2 $\sigma(r_1)\sigma(r_2) = \sigma(r_1 r_2)$